TEMA 2. Sistemas Lineales

Física Computacional I

Carolina Mendoza

Escuela Politécnica Superior
Universidad Nebrija

Dada la matriz

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 4 \\ -1 & 0 & -1 \end{pmatrix}$$

- Hacer un script que sume los elementos de cada fila de la matriz A
- Hacer un script que sume todos los elementos de la matriz A

Este ejemplo suma los elementos de una fila. Puedes modificarlo para resolver los ejercicios anteriores.

```
import numpy as np

A = np.array([[1, 2, 3],
              [2, 3, 4],
              [-1, 0, -1]])
n,m=np.shape(A)
suma=0

for i in range(n):
    suma= suma+A[0,i]
    print(A[0,i])
```

determinar la media de temperatura del fin de semana.

Dada la matriz
Multiplicación de Matrices

```
import numpy as np
np.random.seed(27)
A = np.random.randint(1,10,size = (3,3))
B = np.random.randint(1,10,size = (3,2))
print(f"Matrix A:\n {A}\n")
print(f"Matrix B:\n {B}\n")

D=np.matmul(A,B)
print(D)
C=A@B
print(C)
```

- Generar dos matrices 3×3 con números aleatorios entre 0 y 10.
- Sumar las matrices.
- Multiplicarlas elemento a elemento y mediante producto matricial.
- Calcular el determinante de cada matriz.

- Implementar la serie de Fibonacci hasta un número n dado. Usando un bucle `for` o `while`.

Aplicar la ecuación de caída libre:

$$t = \sqrt{\frac{2h}{g}}$$

- Solicitar la altura h y calcular el tiempo de caída.

- Generar $n = 1000$ valores aleatorios entre 0 y 1.
- Graficar un histograma.

Aplicar la ecuación del movimiento con aceleración constante:

$$x = v_0 t + \frac{1}{2} a t^2$$

- Simular la posición de un objeto en movimiento con aceleración.
- Graficar la posición en función del tiempo.

- Sistemas Lineales. Métodos directos
  - Introducción al problema matemático
  - Clasificación de los métodos numéricos
  - Método de Gauss
  - Factorización LU.

- Resolver un sistema de n ecuaciones con n incógnitas
  Un sistema lineal

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\ \vdots \qquad\qquad \ddots \qquad\quad \vdots &\quad \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n &= b_n \end{aligned}$$

$$\sum_{j=1}^{n} a_{ij}x_j = b_i \qquad i = 1, \cdots, n$$

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} \quad B = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix} \quad X = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$$

$$AX = B \rightarrow AB = \left(\begin{array}{cccc|c} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & b_2 \\ \vdots & \vdots & \ddots & \vdots & \cdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} & b_n \end{array}\right)$$

- Algebraicamente hay que despejar X de la ecuación $AX = B$

$$A^{-1}AX = A^{-1}B \rightarrow X = A^{-1}B$$

- Teorema de Rouché-Frobenius Un sistema de m ecuaciones y n incógnitas es compatible si el rango de A coincide con el de la matriz ampliada.
- La solución es única $rg(A) = n$

- Métodos Directos (Exactos, algoritmos complicados)
  - Gauss
  - Gauss-Jordan
  - Factorización LU
- Métodos Iterativos (Comienzan de una solución inicial, algoritmos sellos, matrices sparse, análizis de convergencia)
  - Jacobi
  - Gauss-Seidel

- Matriz ampliada
- Triangulación y Sustitución

$$\begin{aligned} x_1 + 2x_2 + 3x_3 &= 6 \\ 2x_1 + 3x_2 + 4x_3 &= 9 \\ -x_1 - x_3 &= -2 \end{aligned} \quad \rightarrow AB = \left(\begin{array}{ccc|c} 1 & 2 & 3 & 6 \\ 2 & 3 & 4 & 9 \\ -1 & 0 & -1 & -2 \end{array}\right)$$

Operaciones filas hasta llegar a una matriz diagonal

$$AB = \left(\begin{array}{cccc|c} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & b_2 \\ \vdots & \vdots & \ddots & \vdots & \cdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} & b_n \end{array}\right) \rightarrow$$

$$\overline{AB} = \left(\begin{array}{cccc|c} \overline{a_{11}} & \overline{a_{12}} & \cdots & \overline{a_{1n}} & \overline{b_1} \\ 0 & \overline{a_{22}} & \cdots & \overline{a_{2n}} & \overline{b_2} \\ \vdots & \vdots & \ddots & \vdots & \cdots \\ 0 & 0 & \cdots & \overline{a_{nn}} & \overline{b_n} \end{array}\right)$$

Algoritmo

- Para cada elemento de la diagonal menos el último $k = 0, \cdots, n-1$
- Ceros en las filas bajo la diagonal $i = k, \cdots, n-1$
- Elementos $j = 0, \cdots, n$
- Calcular nuevo elemento a_{ij}

Eliminación hacia adelante

$$A_k = A_k - A_i \frac{a_{k,i}}{a_{i,i}}$$

Sustitución hacia atrás

$$x_i = \frac{b_i^{i-1} - \sum_{j=i+1}^{n} a_{ij}^{i-1} x_j}{a_{ii}^{i-1}}$$

con $i = n-1, n-2, \cdots$

```
matriz pivoteada por fila:
[[ 1.   2.   3.  6.]
 [ 2.   3.   4.  9.]
 [ -1.  0.   -1.  -2]]
```

Fila 1 posición $i = 0$ el elemento $A_{i,i}$ es el pivote.

```
pivote = A[i,i] = AB[0,0] = 1
```

Resto de Filas se referencian con k

```
k = i + 1 = 0 + 1 =  1
```

Calculamos el factor para hacer cero la primera posición

```
factor = AB[1,0]/pivote = 2/1
```

operamos entre la fila k y la fila i, actualizando la fila k

```
        [ 2. 3.  4.  9.]
-(2/1)*[ 1. 2.  3.  6.]

-------------------------
     = [ 0. -1  -2. -3]
```

Nueva matriz AB

```
 AB =
[[ 1.    2.    3.    6 ]
 [ 0.    -1   -2   -3]
 [ -1.    0   -1.    2]]
```

Se repite el procedimiento con

```
k = i + 1 = 2
factor = -1/1
```

```
        [ -1. 0. -1.  -2.]
-(-1/1)*[ 1. 2   3.   6.]
-------------------------
     = [ 0.  2   2.   4]
```

Algoritmo

Nueva matriz AB

```
 AB =
[[ 1.    2.    3.   6]
 [ 0.   -1.   -2   -3]
 [ 0.    2.    2.   4]]
```

Avanzamos a la fila 2

```
i=1
pivote = A[i,i] = AB[1,1]= -1
k = i + 1 = 2
factor = AB[2,1]/pivote  = 2/(-1)
```

```
            [ 0.   2    2.    4.]
-(2/(-1))*[ 0. -1   -2.   -3.]
-------------------------------
     =      [ 0.   0   -2.   -2]
```

Sustitución hacia atrás

```
[[ 1.    2.    3.   6.]
 [ 0.   -1.   -2.  -3.]
 [ 0.    0.   -2.  -2.]]
```

$$x_i = \frac{b_i^{i-1} - \sum_{j=i+1}^{n} a_{ij}^{i-1} x_j}{a_{ii}^{i-1}}$$

Sustitución hacia atrás

```
utlfila = n-1 = 3-1 = 2
ultcolumna = m-1 = 4-1 = 3
```

Última fila

```
[ 0. -2. -2 ]
```

$$-2x_3 = -2$$
$$x_3 = 1$$

Solución:

```
[[1.]
[1.]
[1.]]
```

Una matriz triangular es una matriz cuadrada en la que todos los elementos por encima o por debajo de diagonal principal son cero.
Propiedades:

- el producto de dos matrices triangulares es otra triangular.
- la traspuesta de una matriz triangular superior es una triangular inferior y viceversa.
- El determinante es el producto de los elementos de la diagonal.
- Es invertible si todos los elementos de la diagonal son distintos de cero.
- Los valores propios son los elementos de la diagonal.

- La factorización LU de la matriz es una descomposición en factores de la matriz A de modo que se verifica: $A = LU$ donde:
  - L es una matriz triangular inferior (*Lower*). L matriz triangular inferior con *unos* en la diagonal

$$L = \begin{pmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{pmatrix}$$

  - U es una matriz triangular superior (*Upper*).

$$U = \begin{pmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{pmatrix}$$

Descomponer las siguientes matrices en triangulares:

$$\begin{pmatrix} 2 & 3 & 4 & 5 \\ 6 & 15 & 19 & 23 \\ 8 & 42 & 60 & 70 \\ 12 & 60 & 1 & 17 \end{pmatrix}$$

$$\begin{pmatrix} 2 & 6 & 9 \\ 2 & 5 & 2 \\ -3 & -4 & -11 \end{pmatrix}$$

El método consiste en guardar el valor del pivote en la posición de las filas operadas. $F_2 - 3F_1$ hace que el valor $L_{21} = 3$
Considerando el caso de la matriz

$$A = \begin{pmatrix} 2 & -3 & 5 \\ 6 & -1 & 3 \\ -4 & 1 & -2 \end{pmatrix}$$

Hacemos un cero en la posición fila 1, columna 0.
Para ello, operamos entre la fila 1 y la fila 0, actualizando la fila 1

```
-(6/2)*[ 2 -3  5]
       [ 6 -1  3]
-------------------------
     = [ 0 8  -12]
```

El valor de $l_{1,0} = 6/2 = 3$

Hacemos un cero en la posición fila 2, columna 0.
Para ello, operamos entre la fila 2 y la fila 0, actualizando la fila 1

```
-( -4/2)*[ 2  -3   5]
         [-4   1  -2]

--------------------------
        = [ 0. -5   8]
```

El valor de $l_{2,0} = -2$

Hacemos un cero en la posición fila 2, columna 0.
Para ello, operamos entre la fila 2 y la fila 0, actualizando la fila 2

```
-(-5/8)*[ 0   8  -12]
        [ 0  -5    8]

-------------------------
      = [ 0   0  1/2]
```

El valor de $l_{2,1} = -5/8$

De forma que la matriz

$$A = \begin{pmatrix} 2 & -3 & 5 \\ 6 & -1 & 3 \\ -4 & -1 & -2 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 3 & 1 & 0 \\ -2 & -5/8 & 1 \end{pmatrix} \begin{pmatrix} 2 & -3 & 5 \\ 0 & 8 & -12 \\ 0 & 0 & 2 \end{pmatrix}$$

Convertir el problema $AX = B \rightarrow LUX = B$ en uno de sustitución

- $L(UX) = B$, se denomina $Y = UX$ por lo que se resuelve $LY = B$ determinado Y
- se reemplaza $UX = Y$

Resolver el siguiente sistema de ecuaciones:

$$\begin{aligned} -1.5x_1 + 2.1x_2 + 6.4x_3 &= -8 \\ 2x_1 + 3.9x_2 + 3.1x_3 &= 7.5 \\ 5x_1 + 3x_2 + 2x_3 &= 10 \end{aligned}$$

$$AX = B$$
$$PLUX = B$$

Si $Y = UX$

$$LY = PB$$

```
B1= np.transpose([P@B])
```

```
import numpy as np

A = np.matrix([[2, 3],[1, -2]])
b = np.matrix([[8],[-10]])

if np.linalg.det(A) == 0:
    x = None
    print("No se puede resolver")
else:
    x = (A**-1)*b

print(x)
```

```
import scipy
import numpy as np
A = np.array([[0,1,3], [1,3,-2], [-3,-2,-1]])
LU, piv = scipy.linalg.lu_factor(A)
b = [1, 3, -2]
x = scipy.linalg.lu_solve((LU,piv),b)
print(x)
```

- Método iterativo para resolver sistemas de ecuaciones lineales.
- Se basa en construir una sucesión convergente definida iterativamente.

$$Ax = b$$
$$Dx + Rx = b$$

donde

$$\begin{aligned} A &= D + R \qquad D : \textit{matriz diagonal} \\ R &= L + U \qquad L(U) : \textit{matriz Lower(Upper)} \end{aligned}$$

Solución

$$x = D^{-1}(b - Rx)$$

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ a_{21} & 0 & 0 \\ a_{31} & a_{32} & 0 \end{pmatrix} + \begin{pmatrix} a_{11} & 0 & 0 \\ 0 & a_{22} & 0 \\ 0 & 0 & a_{33} \end{pmatrix} + \begin{pmatrix} 0 & a_{12} & a_{13} \\ 0 & 0 & a_{23} \\ 0 & 0 & 0 \end{pmatrix}$$

$$= \quad \mathrm{L-Lower} \qquad \mathrm{D-Diagonal} \qquad \mathrm{U-Upper}$$

Si $a_{ii} \neq 0$ entonces el método de Jacobi es:
En forma vectorial:

$$x^{(k+1)} = D^{-1}(b - Rx^{(k)})$$

En componentes:

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=0, j\neq i}^{n} a_{ij} x_j^{(k)} \right)$$

El método itera has que dada una tolerancia (tol) se cumple:
En forma vectorial

$$||x^{(k+1)} - x^k|| < tol$$

En componentes

$$\sqrt{\sum_i (x_i^{(k+1)} - x_i^{(k)})^2} < tol$$

Resolver el siguiente sistema de ecuaciones lineales:

$$\begin{aligned} 10x_1 - x_2 + 2x_3 &= 6 \\ -1x_1 + 11x_2 - x_3 + 3x_4 &= 25 \\ 2x_1 - x_2 + 10x_3 - x_4 &= -11 \\ 3x_2 - x_3 + 8x_4 &= 15 \end{aligned}$$

Solución:

$$\mathbf{x} = (1, 2, -1, 1)$$

- Es similar al de Jacobi pero no espera al final de cada iteración para actualizar la variable.
- A medida que se calculan las componentes se usan dentro de la misma iteración.

$$
\begin{aligned}
x_1^k &= \frac{b_1 - a_{12}x_2^{k-1} - a_{13}x_3^{k-1} - a_{14}x_4^{k-1} - \ldots - a_{1n}x_n^{k-1}}{a_{11}} \\
x_2^k &= \frac{b_2 - a_{21}x_1^{k} - a_{23}x_3^{k-1} - a_{24}x_4^{k-1} - \ldots - a_{2n}x_n^{k-1}}{a_{22}} \\
x_3^k &= \frac{b_3 - a_{31}x_1^{k} - a_{32}x_2^{k} - a_{34}x_4^{k-1} - \ldots - a_{3n}x_n^{k-1}}{a_{33}} \\
&= \vdots \\
x_n^k &= \frac{b_n - a_{n1}x_1^{k} - a_{n2}x_2^{k} - a_{n3}x_3^{k} - \ldots - a_{nn}x_n^{k}}{a_{nn}}
\end{aligned}
$$

- Este proceso se puede escribir en forma matricial como:

$$\begin{aligned} a_{11}x_1^k &= b_1 - a_{12}x_2^{k-1} - a_{13}x_3^{k-1} - \ldots - a_{1n}x_n^{k-1} \\ a_{21}x_1^k + a_{22}x_2^k &= b_2 - a_{23}x_3^{k-1} - \ldots - a_{2n}x_n^{k-1} \\ \vdots &= \vdots \\ a_{n1}x_1^k + a_{n2}x_2^k + a_{n3}x_3^k + \ldots + a_{nn}x_n^k &= b_n \end{aligned}$$

Gauss - Seidel de forma Matricial

$$\begin{aligned} (D+L)\mathbf{x}^k &= b-(U)\mathbf{x}^{k-1} \\ &\Downarrow \\ \mathbf{x}^k &= T\mathbf{x}^{k-1}+C \end{aligned}$$

Donde

$$(D+L)^{-1}b = C$$
$$-(D+L)^{-1}U = T$$